1	Create a class Student with StudentId, Name, MobileNo, Address, Course. Write getters and setters for all the data members, Write a method CalculateFee which returns the fee depending on the course taken. Write child classes FastTrackBatchStudent, CorporateBatchStudent, WeekendBatch Student, CorporateWeekendBatchStudent which overrides the CalculateFee method and returns an appropriate fee.You need to identify what are the oops concepts involved. Develop the Java Application

2	Create a class Employee which has the following
	
	int EmpId;
	double Sal = 0;
	double Basic;
	double Allowance;
	double Deductions;
	string FirstName;
	string LastName;
	string Address;
	string Pincode;
	
	Make Salary as ReadOnly
	Write getters and setters real name which returns firstname+lastname
	
	Write a method CalcSalary to calculate salary
	Sal= Basic + Allowance-Deductions
	
	Write a parameterized constructor and save the above values
	
	Write a class PartTimeEmployee extending Employee class and override CalcSalary with different implementation
	
	Write a class FullTimeEmployee extends Employee and override CalcSalary with different implementation
	
	Develop main method class and identify all java OOp's concepts.

3	Create two interfaces IMobile and ITelephone and have common functionalities like Dial in them. Create a class called as Mobile inherited from both the interfaces Create child classes for Mobile - Samsung, Nokia, IPhone which has the following members and Functions Data Members: IEMICode, IsSingleSIM, Processor, SIMCard, MobileNo Member Functions: ConnectBlueTooth, Dial, GetIEMICode, GetWIFIConnection, Receive, SendMessage Generate another level of inheritance like Samsung S5, Nokia Lumis625 and so on...
4	Create a parent class Account which has fields such as AccNo, Name, MobileNumber and methods

	Deposit, Withdraw, GetBalance. Create a child class SavingsAccount where we have interest rate and an extra method AddInterest Create a child class CurrentAccount where we have interest rate and an extra method AddInterest Create a class CheckingAccount where we have an extra member NoOfFreeTransactions, when a transaction is made increment TransactionCount till the number does not exceed NoOfFreeTransactions. If the Count exceeds free transaction then deduct fees from your balance.
5	Write a parent class Fare which has two data members Origin and Destination. Write two methods CalculateFare and PrintFare method which calculates and prints fare respectively. Write child class SeasonalPass - where we have seasonal offer and offers discount of 10% in the fare given Write child class OneTime - where we have no discounts and the fare is the amount to be paid. Write child class FreePass - for freedom fighters -- where they need not pay any amount. Write child class as StudentSeasonalPass inherited from SeasonalPass where they get 30% discount on the fare given. Write child class ChildPass inherited from GeneralPass where they have to take half ticket i.e., 50% of the fare given. Write child class AdultPass inherited from GeneralPass where they have no discount on the fare given. Write child class PhysicallyHandicappedPass inherited from GeneralPass where they get a discount of 40% on the fare given. Write child class SeniorsPass inherited from GeneralPass where they get a discount of 60% on the fare given.
6	Create an abstract class called Vehicle which has a data member -- RegNumber and a abstract method getNumberOfWheels() and a getter for getting Reg Number also override equals method which checks if both the objects are equal depending on regnumber Create a class called TwoWheeler extending Vehicle class This class overrides the method getNumberOfWheels which returns 2; Create a class called ThreeWheeler extending Vehicle class This class overrides the method getNumberOfWheels which returns 3; Create a class called FourWheeler extending Vehicle class This class overrides the method getNumberOfWheels which returns 4; Create a class called Car which extends FourWheeler and the constructor should assign the regNumber

	of the car Create a class called Auto which extends ThreeWheeler and the constructor should assign the regNumber of the Auto Create a class called MotorBike which extends TwoWheeler and the constructor should assign the regNumber of the MotorBike
7	Create a class called Petition which has data members like PetitionId, Name, DateRegistered, Location, City, ProblemDescription, Status. Create two child classes FinancialPetition (Amount Involved) and Non-Financial Petition (CitizenName, ImplementationDate)
8	Create an Advertisment class which has data members like AdvertismentId, AdDescription. Create child classes MatrimonialAdvertisment(Gender,Age,Occupation), RealEstateAdvertisment(RealEstateType,Size,Price)
9	Write a class to handle the basics of a two-player game of Tic-Tac-Toe. Instance Variables: board - a two-dimensional array of chars Turns - an integer keeping track of the number of turns played this game Constructors - TicTacToeClass()the default constructor, which just creates the two-dimensional array and fills each slot with ' ' (a blank space) and initializes the other attributes Accessors boolean isWinner(char p)returns true if the letter passed in currently has three in a row. That is, isWinner('X') will return true if X has won, and isWinner('O') will return true if O has won boolean isFull() - returns true if nine turns have been played and false otherwise boolean isCat() - returns true if all nine spaces are filled AND neither X nor O has won boolean isValid(int r, int c) - returns true if the given row and column corresponds to a valid space on the board int numTurns() - returns the numbers of turns played so far char playerAt(int r, int c)returns the character representing the piece at the given location. Should return either ' ', 'X', or 'O'. void displayBoard() - displays the current board on the screen Modifiers **void** playMove(char p, int r, int c) - allows the given player to place their move at the given row and column. The row and column numbers are 0-based, so valid numbers are 0, 1, or 2